EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

2. Authorization for this examiner's amendment was given in a telephone interview with Mr. Charles Huston (Reg. No. 31,027), on 02/11/2010.

3. Pursuant to MPEP 606.01, the title had been changed to read:

-- SYSTEM FOR DEADLOCK CONDITION DETECTION AND CORRECTION BY ALLOWING A QUEUE LIMIT OF A NUMBER OF DATA TOKENS ON THE QUEUE TO INCREASE --.

4. The specification, page 6, lines 10-15 had been amended as:

An example of a dataflow graph development system is found in U.S. Patent No. 5,999,729. An example of a deadlock resolution system in a multi-threaded environment is found in U.S. Patent No. 6,088,716. Deadlock detection and correction in process networks are known, see, R. Stevens, M. Wan, P. Laramie, T. Parks & E. Lee, Implementation of Process Networks in Java, ~~http://www.ait.nrl.navy.mil/pgmt/ PNpaper.pdf,~~ PNpaper.pdf Juy 1997. An example of a parallel programming environment is found in U.S. Patent No. 6,311,265. All references cited herein are incorporated by reference.

5. The claims had been amended as follow:

Claims 1- 9 (canceled)

Claim 10 (currently amended) A method of deadlock management in a multi-thread, parallel processing data management system having ports for sending and receiving data tokens comprising:

allocating at least one thread to a first process and at least one thread to a second process, wherein the first and second processes are connected through a queue via sending and receiving ports;

using one thread to detect if one or more of said threads are blocked;

if a thread is determined blocked, determining if the blocked thread is sending data or receiving data, wherein a receiving port of said blocked thread blocks if a data token is unavailable and a sending port of said blocked thread blocks when a queue limit is reached; ~~and~~

determining if a deadlock exists using said block detecting thread by building a wait graph of said one or more blocked threads and determining if the graph is cyclic, wherein if said graph is cyclic, said graph is waiting on itself, indicating ~~a~~ the deadlock exists; and

correcting the deadlock, if the deadlock is detected, by allowing the queue limit of a number of the data tokens on the queue to increase.

Claim 11 (currently amended) The method of claim 10, blocking ~~a~~ the receiving port when a data token is not available.

Claim 12 (currently amended) The method of claim 10, blocking ~~a~~ the sending port when a limit on ~~the~~ a number of data tokens in the queue is reached.

Claim 13 (currently amended) The method of claim 10, including building ~~a~~ the wait graph with said blocked threads and traversing said wait graph to determine if it is cyclic.

Claim 14 (canceled)

Claim 15 (currently amended) The method of claim ~~14~~ 10, wherein the limit of a queue associated with a sending port is allowed to increase.

Claim 16 (currently amended) The method of claim ~~14~~ 10, wherein a queue limit on the number of data tokens of a second queue is decreased while said limit of said ~~first~~ queue is increasing.

Claims 17- 20 (canceled)

Claim 21 (currently amended) A method for executing a dataflow application comprising:

providing a dataflow application comprising a plurality of map components and data ports, some of said map components being linked between data ports and some map components

comprising one or more composite components having a plurality of processes, wherein at least some of said linked data ports being linked by a queue;

allocating a processing thread to a respective map component;

executing multiple processing threads in parallel with each map component on a separate processing thread;

using a thread to detect if a deadlock condition does or will exist for one or more of said processing threads by building a wait graph of several thread states and determining if the wait graph is circular; and

correcting a deadlock condition for a deadlocked processing thread by allowing ~~a first~~ the queue linking data ports to exceed a queue limit of a number of data tokens.

Claim 22 (previously presented) The method of claim 21, wherein the correcting step includes choosing a thread that waits as a producer if a circular wait graph is detected.

Claim 23 (currently amended) The method of claim 21, wherein if the detecting step determines a wait graph is circular, the correcting step including analyzing queues other than said ~~first~~ queue in the wait graph for token batch reduction.

Claim 24 (currently amended) The method of claim 21, wherein if the detecting step determines a wait graph is circular, the correcting step including the substep of reducing said queue limit in one or more queues other than said ~~first~~ queue in the wait graph.

Claim 25 (currently amended) A method for executing a dataflow application in a multi-thread processing system comprising:

~~a)~~ providing a dataflow application comprising a plurality of map components and data ports, a number of map components being linked between data ports using queues and some map components comprising composite components having a plurality of processes;

~~b)~~ allocating a processing thread to each composite map component including allocating a thread for deadlock detection;

~~c)~~ executing each composite map component on a separate thread; ~~and~~

~~d)~~ determining if a deadlock exists using said deadlock detection thread to monitor queues, including building a wait graph and determining if the graph is cyclic, ~~and if a deadlock exists, allowing a queue limit to increase~~ wherein if said graph is cyclic, said graph is waiting on itself, indicating the deadlock exists [[.]] ; and

correcting the deadlock, if the deadlock exists, by allowing a queue limit of a number of data tokens on the queues to increase.

Claim 26 (canceled)

Claim 27 (currently amended) The method of claim ~~26~~ 25, wherein the queues including a first queue and a second queue, and a queue limit on the number of data tokens of the second queue is decreased while ~~said~~ a limit of said first queue is increasing.

Claim 28 (previously presented) The method of claim 25, including transporting data tokens among map components on said queues.

Claim 29 (previously presented) The method of claim 28, batching the data tokens to regulate the length of time a map component may execute without synchronization.

Claim 30 (previously presented) The method of claim 25, including ports associated with each map component for representing and transporting multi-state null value tokens.

Claim 31 (previously presented) The method of claim 30, the null value tokens including an error null.

Claim 32 (previously presented) The method of claim 1, said block detection thread monitors one or more of the data queues.

Claim 33 (previously presented) The method of claim 32, said block detection thread does not require communication from other threads to determine if a thread is blocked.

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jennifer N. To whose telephone number is (571) 272-7212. The examiner can normally be reached on M-T 6AM- 3:30 PM, F 6AM- 2:30 PM.

7. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

8. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Meng-Ai An/
Supervisory Patent Examiner, Art Unit 2195

/Jennifer N To/
Patent Examiner, AU 2195